# ホックニーモデルを用いたベクトル計算機の性能評価その応用(スーパーコンピュータのための数値計算アルゴリズムの研究)

AUTHOR(S):

島崎, 眞昭

CITATION:

島崎, 眞昭. ホックニーモデルを用いたベクトル計算機の性能評価その応用(スーパーコンピュータのための数値計算アルゴリズムの研究). 数理解析研究所講究録 1987, 613: 109-120

ISSUE DATE:

1987-03

URL:

http://hdl.handle.net/2433/99796

数理解析研究所講究録
613巻 1987年 109-120



# ホックニーモデルを用いたベクトル計算機の性能評価その応用

京大大型計算機センター　島崎　眞昭（Ｍａｓａａｋｉ　Ｓｈｉｍａｓａｋｉ）

## 1. はじめに

スーパーコンピュータの国産化を契機として、我国でもスーパーコンピュータが大学の全国利用大型計算機センターや国立研究機関に導入され、また産業界にも普及する傾向がみられ、本格的な超大型計算の時代が始まったといえる。現在のスーパーコンピュータは演算パイプライン方式ベクトル計算機〔１〕で、ピーク性能は高いが、その高速演算性能を十分発揮させるのは必ずしも容易でない。実際レジスタ・レジスタ型のベクトル計算機では、三種の典型的な実効性能があるといわれる〔２〕。

(1) スカラ性能

ベクトル演算機能が使用されない場合の性能

（ＣＲＡＹ－１では０－４ＭＦＬＯＰＳ）

(2) ベクトル性能

ベクトル化が行なわれて、パイプライン演算器の性能が発揮される場合の性能

（ＣＲＡＹ－１では４－５０ＭＦＬＯＰＳ）

(3) スーパーベクトル性能

ロード／ストアの回数を減少させるために、ベクトル・レジスタを有効に使用し、チェイニングが効果的に使用する場合の性能

（ＣＲＡＹ－１では５０―１６０ＭＦＬＯＰＳ）

この三種の性能の違いが大きく、かつアルゴリズムやプログラムの少しの変化で性能が例えば、(2)から(3)へ改善出来る場合もあるので、ベクトル計算機の性能評価は容易でない。このため、性能の検討には、ハードウェア／ソフトウェアのカタログ性能の検討の他、短い核ループ・プログラムによる性能評価および、実用規模の応用プログラムによるベンチマークテストが行なわれる。ベクトル計算機は機種毎の個性も強く、統一的なモデル化、性能指標の設定が容易でないため、計算機利用の立場では、有効利用のための機種毎のノウハウの追求に走る傾向がある。しかし限定された形であっても、モデルによる性能の定量評価を試み、計算機アーキテクチャとアルゴリズムの性能との関係を少しでも明確化することが必要と考えられる。

本稿では、主としてＨｏｃｋｎｅｙのモデルを用い、計算機の演算性能の評価とその応用について検討する。

## 2. ベクトル計算機のモデル化

### 2.1 Ｈｏｃｋｎｅｙのモデル

パイプライン演算器のピッチ時間をｔｐ、演算開始準備時間をｔｉ、演算器のステージ数

を$\ell$とする。ベクトル命令起動後最初のｔｉ＋$\ell$ｔｐの間は結果が生成されないが、その時間以後はｔｐ毎に一つの演算結果が得られるとするとき、ｎ個の演算を行うのに必要な時間ｔは次式で与えられる。

$$t = t_i + \ell t_p + (n-1) t_p \qquad (1)$$

ｔの単位を$\mu$ｓｅｃとしたとき、演算速度　をＭＦＬＯＰＳ（Ｍｉｌｌｉｏｎｓ　ｏｆ　Ｆｌｏａｔｉｎｇ　Ｐｏｉｎｔ　Ｏｐｅｒａｔｉｏｎｓ　ｐｅｒ　Ｓｅｃｏｎｄ）で表したものは次式で与えられる。

$$\upsilon = n / t$$

$$= t_p^{-1} \left( 1 - \frac{1}{n} + \frac{\ell}{n} + \frac{1}{n} \frac{t_i}{t_p} \right)^{-1} \qquad (2)$$

ここではｎはベクトル・レジスタの要素数　より小さいとする。$n > n_r$のときの影響については後に述べる。

Ｈｏｃｋｎｅｙ〔３〕はベクトル計算機やプロセッサ・アレイの特徴を把握するための特徴パラメータを提案している。それらはベクトル計算機の高速化の方式と、その影響やベクトル計算機向き数値解法の性能を考えるときに有効であると考えられるので、ここでもそれを用いることにする。Ｈｏｃｋｎｅｙは一般に演算の個数ｎと計算時間ｔの関係を

$$t = r_\infty^{-1} ( n + n_{1/2} ) \qquad (3)$$

で表現し、$r_\infty$、$n_{1/2}$を計算機の特徴パラメータとすることを提案している。

式(3)より、演算速度　は

$$\upsilon = r_\infty ( 1 + n_{1/2} / n )^{-1} \qquad (4)$$

となり、ｒ　は最大性能と呼ばれ、ｎ→　のとき達成される漸近的性能である。

式(4)よりわかるように、$n = n_{1/2}$のとき最大性能の½の速度が達成され、$n_{1/2}$は半性能長と呼ばれる。式(3)より、ｎ－ｔのグラフがｎ軸を$-n_{1/2}$で切ることがわかる。

パイプライン演算器については、(1)、(2)と(3)、(4)とから，

$$r_\infty = t_p^{-1} \qquad (5)$$

$$n_{1/2} = \ell - 1 + t_i / t_p \qquad (6)$$

となる。

## 2.2　多重化パイプライン演算器と実効性能

現用のベクトル計算機では最大性能を高くするため、(1)独立した複数パイプライン（同種演算の場合を含む）の設置とその並行動作方式、または(2)パイプライン演算器の多重化方式が用いられる。後者においては、パイプライン演算器のハードウェアをたとえば２重化し、ベクトル・データの奇数番目の要素は第１のパイプライン演算器で、偶数番目の要素は第２のパイプライン演算器で計算する方式が用いられる。一つのベクトル命令で、２本のパイプラインが起動され、データの振り分けは、ハードウェアにより自動的に行われるので、いわば太い１本のパイプラインとして動作し、データ配分方式とも呼ばれる。ｍ本のパイプラインが多重化された場合を考える。簡単のためｍｏｄ（ｎ，ｍ）＝０の場合を考える。この場

合の計算時間tは式(1)と若干異なるだけで、

$$t = ti + \ell tp + (n/m - 1) \cdot tp \qquad (7)$$

となり、

$$r_\infty = m \cdot tp^{-1} \qquad (8)$$

$$n_{1/2} = m \{ (\ell - 1) + ti/tp \} \qquad (9)$$

となる〔1〕。ただしmod(n, m) ≠ 0のとき、式(7)～(9)は若干変化する。式(8)、(9)より、最大性能長と半性能長はともにm倍となる。n－tのグラフ上ではt軸を切る点は不変で、直線の傾きが1／mとなり、n軸を切る点が原点よりm倍離れることからもわかる。スカラ処理に対するn－tの直線とベクトル処理に対するn－tの直線との交点のnの値が、限界ベクトル長または交叉ベクトル長であるが、パイプラインの多重化により、限界ベクトル長は若干短かくなることがわかる。

パイプラインの多重化により、最大性能はm倍になるが、ベクトル長が有限のとき、性能の変化がどのようになるか検討してみる。パイプラインが1本のときとm本のときの、演算速度をそれぞれ $v_1$、$v_m$とすると上述の議論より、

$$v_1 = r_\infty (1 + n_{1/2}/n)^{-1} \qquad (10)$$

$$v_m = m r_\infty (1 + m n_{1/2}/n)^{-1} \qquad (11)$$

と書ける。速度向上比Rmを次式で定義する。

$$Rm = \frac{v_m}{v_1}$$

$$= m(1 + x)/(m + x) \qquad (12)$$

ただし$x = n/n_{1/2}$。xはベクトル長を1本のパイプラインに対する半性能長で正規化したものといえる。m＝2，4に対するx－Rmのグラフを図1に示す。このグラフは有限のベクトル長に対し、mを1より増加させたとき、実効的に性能がどの程度向上するのか目安を示すものといえる。このグラフからわかるように、m＞1にしたとき、Rm＞1で性能が低下することはない。x＝1に対しては、Rm≒1.3即ち性能は3割増しにすぎない。x＝2即ちm＝2の場合の半性能長のベクトル長に対する速度はm＝1に対して、Rm＝1.5即ち性能が1.5倍になることがわかる。m＝2のとき、Rmが1.8以上になるためには、x＝n／$n_{1/2}$＞8以上でなければならないことがわかる。

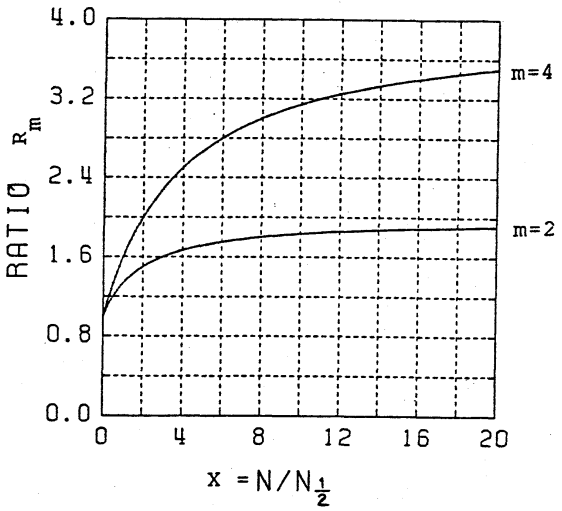



図1　ベクトル長と多重化パイプライン演算器の実効性能

有限のベクトル長に対して、m＞1の効果を発揮するためにはx＝n／$n_{1/2}$＞＞1が必要であるが、nは解くべき問題やアルゴリズムにより決定され、いくらでも大きくできる訳ではない。従って、計算機として$n_{1/2}$が小さいことが大切であることがわかる。またプログラムからみた$n_{1/2}$を小さくすることも有効である。

## 2.3 FORTRANプログラムによる性能パラメタの評価

基本的演算の計算時間やHocknyの$r_\infty$，$n_{1/2}$等はFORTRANプログラムで計測することも可能で、FACOM VP 100の結果については、文献〔1，4〕に与えた。図2に、VP 100とVP 200に対する計測結果の一例を示す。

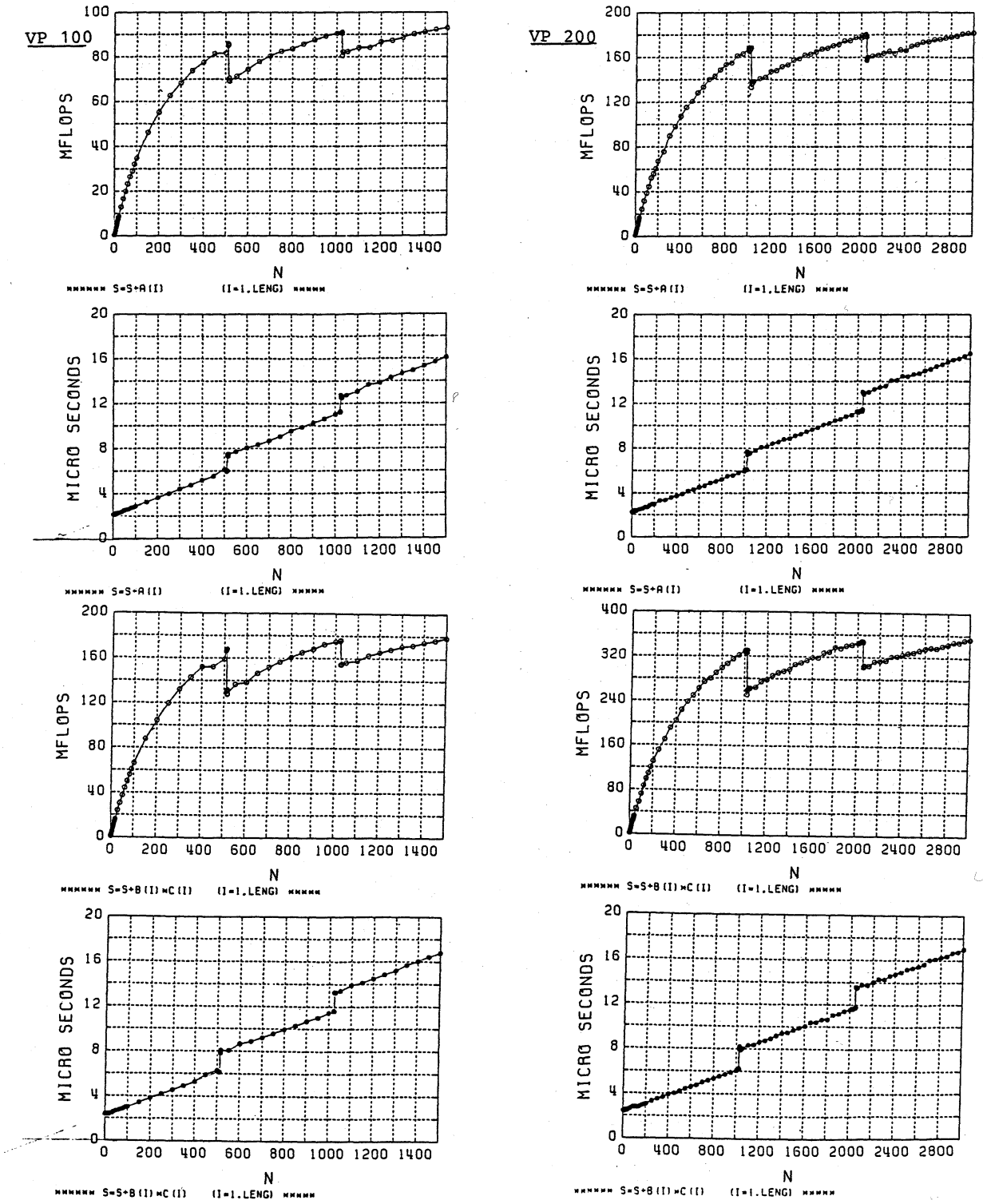



図2 FACOM VP 100／200 に関する計算速度の計測例

注意すべき点は(1)グラフの形は同一、(2)ＶＰ２００のｎ軸の目盛はＶＰ１００のそれの２倍、(3)ｔ軸の目盛は同一、(4)ＶＰ２００のＭＦＬＯＰＳ軸の目盛はＶＰ１００のそれの２倍、となっている。このことは2.2の議論が実際に当てはまることを示し、ＶＰ２００の半性能長はＶＰ１００の２倍になることがわかる。

ベクトル・レジスタの要素数 $n_r$ より大きいベクトル長に対する演算は $n_r$ 毎に区切って行うことになるが、この繰り返し制御をコンパイラの生成するスカラ命令で行うシステムではｎ－ｔ直線は一般には、図２に示すように階段的になる。ｎ－ｔのグラフのｔ軸切片 $t_0$ にはコンパイラの生成するループ制御命令の寄与する部分が大きい。ＦＡＣＯＭ ＦＡＣＯＭ ＦＯＲＴＲＡＮ ７７／ＶＰコンパイラでは、ｎが $n_r$ 以下であるとコイパイラが判定できる場合、不要の命令の生成が抑止され、$t_0$ が半分程度になることもある。本来立上がり時間の長い総和演算ではほとんど変化しない。

$t_0$ の増加は $n_{1/2}$ を増加させる[1]。代入文の右辺の項の数が多い場合やＤＯ文の内部の文の数が多いと、ループ制御に伴う $t_0$ の寄与はいわば多数の演算に分配され、結果として、ループの $n_{1/2}$ が減少し、立上りが良くなる。

以上のようにＦＯＲＴＲＡＮで計測される $n_{1/2}$ はハードウェアのもつ $n_{1/2}$ というより、ハードウェアとプログラムとを一体として考えたときの $n_{1/2}$ となる。パラメータのプログラム依存性が生ずるが、実際に観測される現象の説明・理解には有用であるといえる。

## 2.4 計算機のマクロモデルと実効性能

ジョブ全体をスカラ処理したときの全計算量のうち、Ｖの割合がベクトル処理可能のときベクトル化率をＶで定義する。ベクトル処理を行うために必要となる余分のスカラ命令の実行時間、言わば雑時間やベクトル処理ユニットとスカラ処理ユニットが時間的に重なって並行動作する時間は全体の計算時間に比べ無視できる程度に小さいとする。

ジョブ全体をスカラ処理ユニットだけで実行したときの計算時間を $T_S$、ベクトル処理ユニットをも用いて処理したときの計算時間を $T_V$ とし、$T_V$ のうちベクトル処理ユニットの処理時間を $T_{VU}$ とすると、次式をうる。

$$T_V = (1-V)\cdot T_S + T_{VU} \qquad (13)$$

ベクトル処理ユニットの計算速度はスカラ処理ユニットの計算速度の $\alpha$ 倍とし、実効性能向上比 Ｅを $T_S / T_V$ で定義すると

$$\alpha = V T_S / T_{VU} \qquad (14)$$

$$E = \frac{1}{1 - V + V/\alpha} \qquad (15)$$

式（１５）より、Ａｍｄａｈｌの法則として知られているように、Ｖが相当１に近くない限り、$\alpha$ が大きくてもＥはあまり大きくならないことがわかる。$\alpha$ はベクトル長やベクトル演算器の利用率に依存する。スカラ処理ユニットでは、計算速度はベクトル長には殆ど依存しないので、ベクトル長ｎと $\alpha$ の関係は図２に示したｎとＭＦＬＯＰＳ値の関係に相似となる。

2．2で多重化パイプライン演算器の実効性能について検討した。そこではベクトル演算器単体の性能を問題にしたので、計算機としてはベクトル化率V＝1の場合を対象としたことになる。多重化パイプライン演算器の実効性能に対するベクトル化率の影響を見ることにする。

まず最初にベクトル長は十分長いが、ベクトル化率は必ずしも1でない場合についてm本のパイプラインの多重化の効果について考える。単一パイプライン演算器のスカラ演算器に対する計算速度比を$\alpha_1$とし、m本の多重化パイプライン演算器については$m\alpha_1$とする。そのときの式（15）のEをEmと表すと

$$Em = \frac{1}{1 - V + V / (m\alpha_1)} \qquad (16)$$

EmとE1（m＝1）との比REを相対性能向上比と定義すると、次式を得る。

$$RE = \frac{Em}{E1} = \frac{1 - V + V/\alpha_1}{1 - V + V/(m \cdot \alpha_1)} \qquad (17)$$

図3にm＝2の場合で、$\alpha_1$をパラメータとしたときのV－REの曲線を示す。$\alpha_1$が大きくなると、REをmに近ずけるためには、Vがより1に近ずかなければならないことが分る。

次にベクトル長が有限の場合を考える。ここで

$$\alpha_1 = \alpha_\infty (1 + x^{-1})^{-1}, \qquad x = n / n_{1/2} \qquad (18)$$

と仮定し、式（18）を式（17）に代入すると、REと$n_{1/2}$で正規化されたベクトル長xとの関係がわかる。図4にm＝2，$\alpha_\infty$＝15の場合でVをパラメータとしたx－REの曲線を示す。

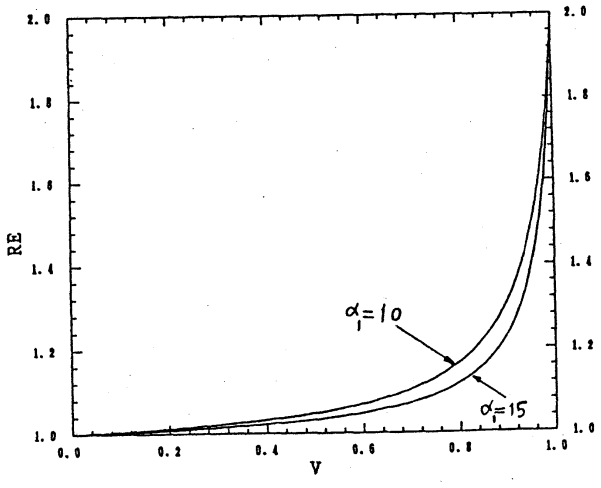



図3　ベクトル化率と相対性能向上比

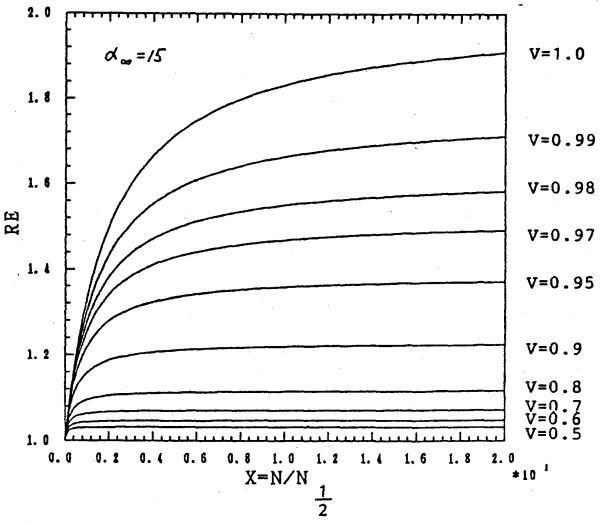



図4　ベクトル長と相対性能向上比

実際のプログラムについて、ベクトル化率やαの値を正確に実測することは必ずしも容易ではない。αは演算器の速度だけで決るのでなく、ベクトル・レジスタを活用し、ロード／ストア命令を減少させ、ベクトル演算命令をできるだけ時間軸上密に実行させることにより実効的に大きくできる。これについては3で検討する。

適当なソフトウェアツール例えばＦＡＣＯＭ　ＶＰの場合には、会話型ベクトライザーの動的解析を用いるとベクトル長やベクトル化率および実効性能向上比Ｅの値の予測値がえられる。一つのジョブについてスカラー・モードおよびベクトル・モードで実行し、Ｔｓ，Ｔｖ，Ｔｖｕを測定し、かつベクトル処理ユニットとスカラー処理ユニットの処理が時間的に重なっていないと仮定できるとすると、式（13）が使えて

$$V = \{Ts - (Tv - Tvu)\} / Ts \qquad (19)$$

よりベクトル化率が実測できる。Ｔｖｕ／ＴｖをＶＵ率といい、βで表す。

$$\beta = Tvu / Tv \qquad (20)$$

α，β，Ｅ，Ｖについて次の式も有用である。

$$\alpha = (E - 1) / \beta + 1 \qquad (21)$$

$$V = \alpha \cdot \beta / E \qquad (22)$$

幾つかの実例のプログラムについて、Ｖ，Ｅの実測値と予測値との比較検討した結果については文献〔5〕を参照してほしい。なお京都大学大型計算機センターのジョブについて見るとβは約５０％程度である。

## 3.　数値計算アルゴリズムの性能評価への応用

### 3.1　ベクトル計算機上での数値計算アルゴリズム評価の問題点

数値計算アルゴリズムの評価はｉ）所要計算時間，ｉｉ）誤差および安定性等について行なわれるが、ここでは、所要計算機時間に話を限定する。従来からの逐次処理型の汎用計算機に対しては、所要計算時間の評価は所要計算量の評価としてとらえられ、必要な四則演算の個数の評価で行なわれてきたし、それは実際に必要な計算時間の評価に有用であった。しかしベクトル計算機上での必要計算時間は、すでに見たように計算機アーキテクチャとの適合性さらにプログラムにも大きく依存する。現用のベクトル計算機の個性は強く、すべてを統一的にとらえるようなモデルの設定は容易でなく、計算機毎に定性的にあるいはノウハウ的に有効利用の方法が述べられることが多いが、可能な限り共通なモデルに基づき、かつ出来れば定量的にアルゴリズムの評価を行うことが望ましい。さて既に述べたように、ベクトル計算機の実効性能にベクトル長が大きく影響する。数値計算アルゴリズムを考えてみると、大型疎行列問題に対する解法のように、通常使用される場合、問題のサイズが十分大きく、アルゴリズムがベクトル化可能であれば、ベクトル長は十分長く、ベクトル長の影響を考えなくてよいものと、有限のベクトル長の影響が重要となるものがある。前者については、計算に必要なサイクル数の評価を行えばよいし、後者については、より細かなモデルで議論する必要がある。本稿では後者の例として、行列積演算をとりあげることにする。疎係数行列

連立一次方程式に対する、ベクトル計算機向きＩＣＣＧ法は前者の例として　扱えるが紙数の関係もあり、文献〔８〕に譲る。

### 3.2　行列積演算アルゴリズムの評価

密行列の積演算を考える。行列積演算は極めて単純な演算であるが、その計算には通常 $2n^3$の浮動小数点演算が必要であり、連立一次方程式の解の２／３ $n^3$より必要演算量が多い。行列積演算の基本的重要性から、既にプログラムとして高速のものが開発されているので、ここで述べるアルゴリズムは新しいものではないが、プログラミングのノウハウとして知られているものを、モデルに基づき、解析、評価することに意義があると考えられる。

行列積演算は一般的に

```
for ―― =1 to ――
 for ―― =1 to ――
  for ―― =1 to ――
     a_ij = a_ij + b_ik * c_kj
  end
 end
end
```

の形に書ける。ここでループ制御変数は意図的に省かれている。ｉ、ｊ、ｋをどの順で制御変数にするかにより、次の６通りが可能である。

(1)内積型：ｉｊｋ*、ｊｉｋ　(2)中間積型：ｉｋｊ、ｊｋｉ*　(3)外積型：ｋｉｊ、ｋｊｉ*

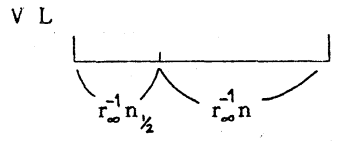
最内側ループのメモリアクセスを連続アクセスにするのが効率がよく、＊を付してある。ＣＲＡＹ－１やＦＡＣＯＭ　ＶＰ（以下では、ＦＡＣＯＭ　ＶＰはロード／ストア・パイプ２本を有するＦＡＣＯＭ　ＶＰ　100／200 の意味で用いる）では中間積の効率が良いことが知られている〔６、７〕。松浦、三浦〔６〕はＨｏｃｋｎｅｙのモデルを用いて、中間積法が、内積法より効率のよいことを示した。その結果によれば、$n \ll n_{1/2}$で速度比が大きく、$n \gg n_{1/2}$で両者の差が小さくなる。ここで、ロード／ストアの影響、ベクトル・レジスタのリストアの影響は除外されている。中間積法の第２のループにループアンローリング手法を適用すると、効果の大きいことがノウハウとして知られているが、これについて考えるには、さらに詳細な計算機モデルを仮定する必要がある。Ｄｏｎｇａｒｒａ等〔７〕の用いたものに近いＣＲＡＹ型のものとして、仮想の次のベクトル命令を考える。

| | | |
|---|---|---|
| ＶＬ | : Vector Load | メモリからベクトルレジスタへのロード<br>定数ベクトルのロードも含めて考える |
| ＶＳＴ | : Vector Store | ベクトルレジスタからメモリへのストア |
| ＶＡＤ | : Vectot Add Vector | ベクトルの要素同志の加算 |
| ＶＭＶ | : Vector Multiply Vector | ベクトルの要素同志の乗算 |
| ＶＭＳ | : Vector Multiply Scalor | ベクトルの各要素にスカラを掛ける |
| ＶＳＵＭ | : Vector Sum | ベクトルの要素の総和演算 |
| ＶＴＲ | : Vector Register Transfer | ベクトルレジスタからベクトルレジスタへの転送 |

ベクトル計算機では汎用機と異なり、一般に被演算子用レジスタと演算結果用レジスタを共用させることができないので、ＶＴＲ命令を仮定した。

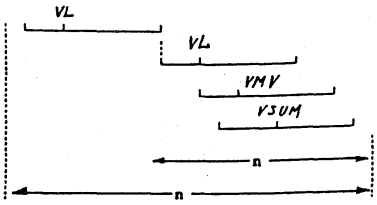
簡単のため全命令は１チャイムに１個の結果を生成でき、$n_{1/2}$はＶＳＵＭ命令を除き等し

く、ＶＳＵＭ命令の半性能を$n'_{1/2}$（$>n_{1/2}$）とする。Ｄｏｎｇａｒｒａに従い命令実行のタイムチャートを次のように表す。



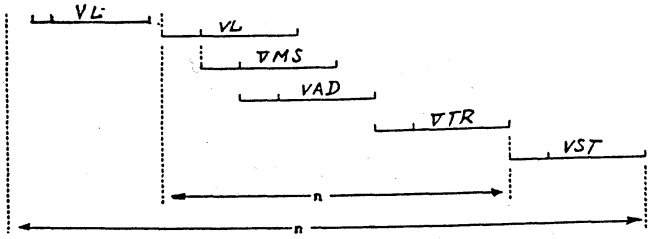

行列積演算のプログラムとそれに対応して考えられる命令実行のタイムチャートを示した。ただし、ここではセカンダリーループのチェイニングはない場合のみを扱うことにした。

ｉ）内積型（ｉｊｋ）



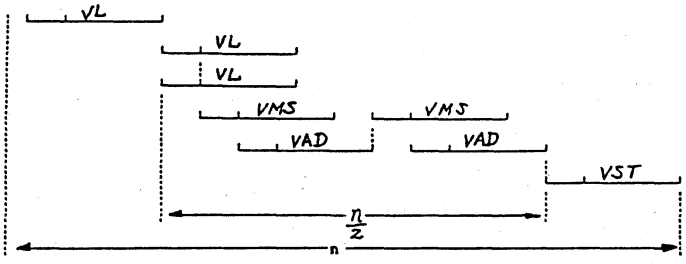

$$t_{ijk} \doteq r_\infty^{-1} n \cdot \{ n \cdot ( n + 2 n_{\frac{1}{2}} + n'_{\frac{1}{2}} ) + n + n_{\frac{1}{2}} \}$$

ⅱ）中間積型（ｊｋｉ）ただしアンローリングを行なわない場合




$$t_{mdl} = r_\infty^{-1} \cdot n \cdot \{ n \cdot ( 2 n + 4 n_{\frac{1}{2}} ) + 2 n + 2 n_{\frac{1}{2}} \}$$
$$= 2 r_\infty^{-1} n \cdot \{ n ( 2 n + 2 n_{\frac{1}{2}} ) + n + n_{\frac{1}{2}} \}$$

ⅲ）中間積型（ｊｋｉ）＋アンローリング




$$t_{mdunrol} = r_\infty^{-1} n \cdot \{ \frac{n}{2} ( 2 n + 5 n_{\frac{1}{2}} ) + 2 n + 2 n_{\frac{1}{2}} \}$$

このモデルにおけるアンローリングの効果を$r_u = t_{mdl} / t_{mdunrol}$で定義すると

$$r_u = \frac{2\{n(n+2n_{\frac{1}{2}}) + n + n_{\frac{1}{2}}\}}{\{n(n+\frac{5}{2}n_{\frac{1}{2}}) + 2n + 2n_{\frac{1}{2}}\}}$$

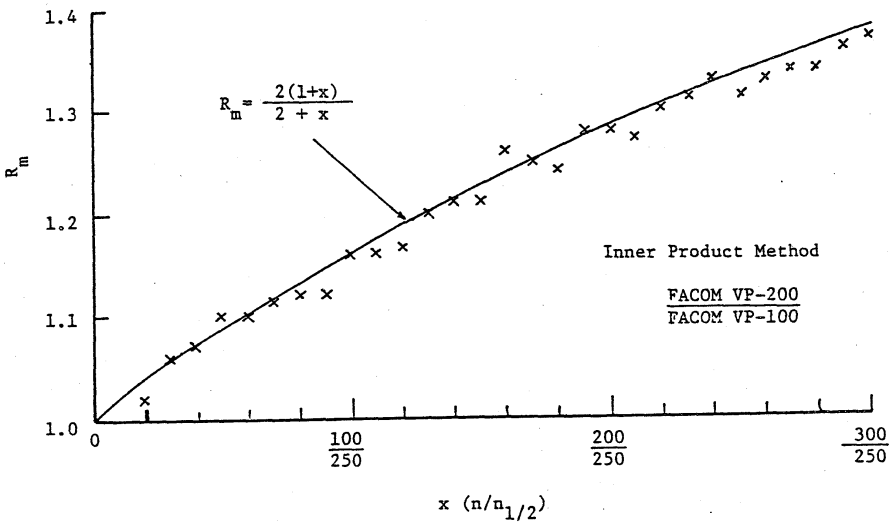
$n \gg \sqrt{n_{\frac{1}{2}}}$なる場合に二次の項のみ考えると

$$r_u = \frac{2(x+2)}{x+5/2} = 2 - \frac{2}{2x+5} \longrightarrow 2 \quad (x \gg 1)$$

ｒｕは２より小さい方から２に漸近する。

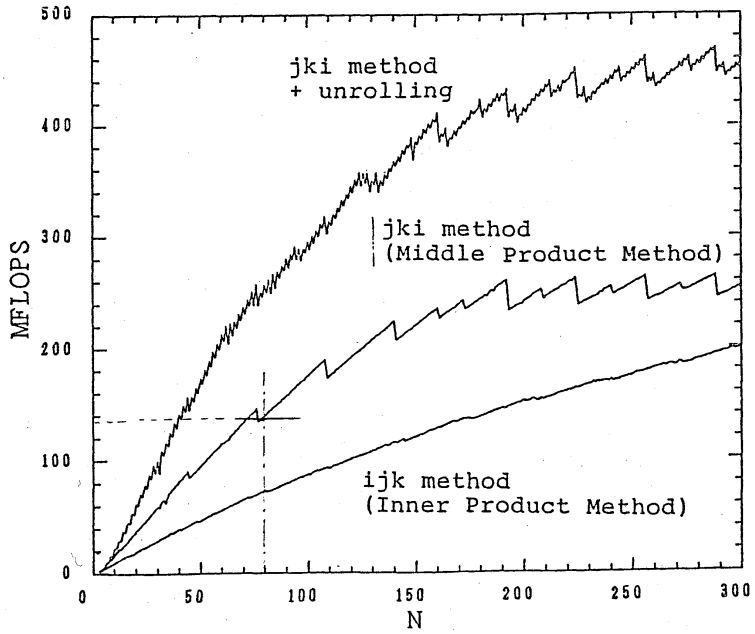
京都大学大型計算機センターのＦＡＣＯＭ　ＶＰ　２００／１００についての、行列積演算の速度の実測例を示す。図５にＦＡＣＯＭ　ＶＰ　２００（ｍ＝２），ＶＰ　１００（ｍ＝１）の速度比の実測値と、ベクトル化率　Ｖ＝１として　Ｒ　＝２（１＋ｘ）／（２＋ｘ）のグラフを示す。Ｎ　として、図２より近似的に求めた概略値Ｎ　＝２５０を用いている。図６に　ＦＡＣＯＭ　ＶＰ　２００において計測した３種のアルゴリズムのＭＦＬＯＰＳ値を示す。この計測に於て、計測に不都合なコンパイラの最適化が作用するのを防ぐため、意図した最内側ループのみベクトル化されるようにコンパイラ・オプションを指定した（コンパイラ：ＦＯＲＴＲＡＮ　７７／ＶＰ　Ｖ１０Ｌ２０）。図７には中間積演算に於けるループ・アンローリングの効果について、$r_u$ の実測値と近似式による結果の比較を示している。Ｎ　としては、図６より近似的に求めた概略値８０を用いている。$r_u$はｎの増加とともに約１．６から２に漸近していく。

ここで定義したモデルが実際の多くのベクトル計算機に適合するかは、機種毎に検討する必要がある。実際には、セカンダリー・ループにおけるチェイニングの影響や命令毎のｎの違いの影響に対する配慮も必要であろう。




Test Matrix : Frank Matrix, 64 bits
Measurement FACOM VP-l00 FORTRAN 77/VP Vl0Ll0 '85 6.18
FACOM VP-200 FORTRAN 77/VP Vl0L20 '85 ll.l8

図５　ＦＡＣＯＭ　ＶＰ　２００／１００における
内積法による行列積演算の計算速度の比較




テスト行列 Frank行列 64ビット

図6 FACOM VP 200における行列積演算の計算速度

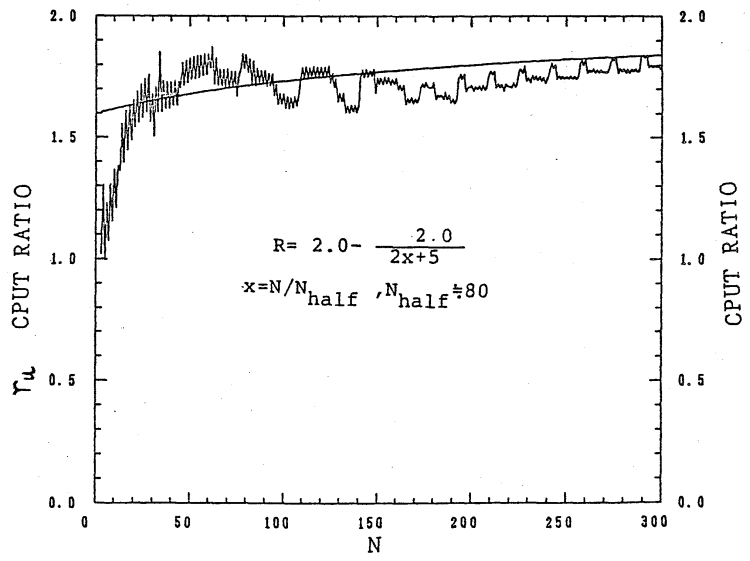



図7 中間積法におけるループ・アンローリングの効果

## 5. むすび

スーパーコンピュータの演算性能評価とアルゴリズムの評価について若干の検討を試みた。極めて限定された問題しか扱えなかったが、今後種々のアルゴリズムについて、評価と実測される計算速度との対照・比較検討した事例を蓄積してアルゴリズムと計算機アーキテクチャの関係の明確化を進めることが重要と考えられる。

## 参考文献

[1] 島崎真昭：スーパーコンピュータ入門、コンピュータソフトウェア、Vol.2, No.3(1985), pp.2-26.

[2] J.J. Dongarra, S.C. Eisenstat: Squeezing the Most out an Algorothm in CRAY FORTRAN, ACM Trans. Mathematical Software, Vol.10, No.3(1984), pp.219-230.

[3] R.W. Hockney, C.R. Jesshope: Parallel Computers, Adam Hilger, 1981.
奥川峻史、黒住祥佑共訳：並列計算機、共立出版、１９８４.

[4] 島崎真昭：京都大学大型計算機センターのベクトル計算機システムFACOM VP 100とある種の線形計算ライブラリーのベクトル化について、情報処理学会数値解析研究会資料、数値解析12—1、1985.2.22.

[5] 戸田昇、島崎真昭：京都大学大型計算機センターのスーパーコンピュータの利用実績の分析と考察、情報処理学会第３２回全国大会講演論文集、４Ｗ－10、1986.

[6] J.J. Dongarra, F.G. Gustavson, A. Karp: Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine, SIAM Review, Vol.26, No.1(1984), pp.86-92.

[8] 島崎真昭：スーパーコンピュータの演算性能の評価とアルゴリズムの評価、情報処理学会「コンピュータ・システム」シンポジウム、pp.19-28, 1985.